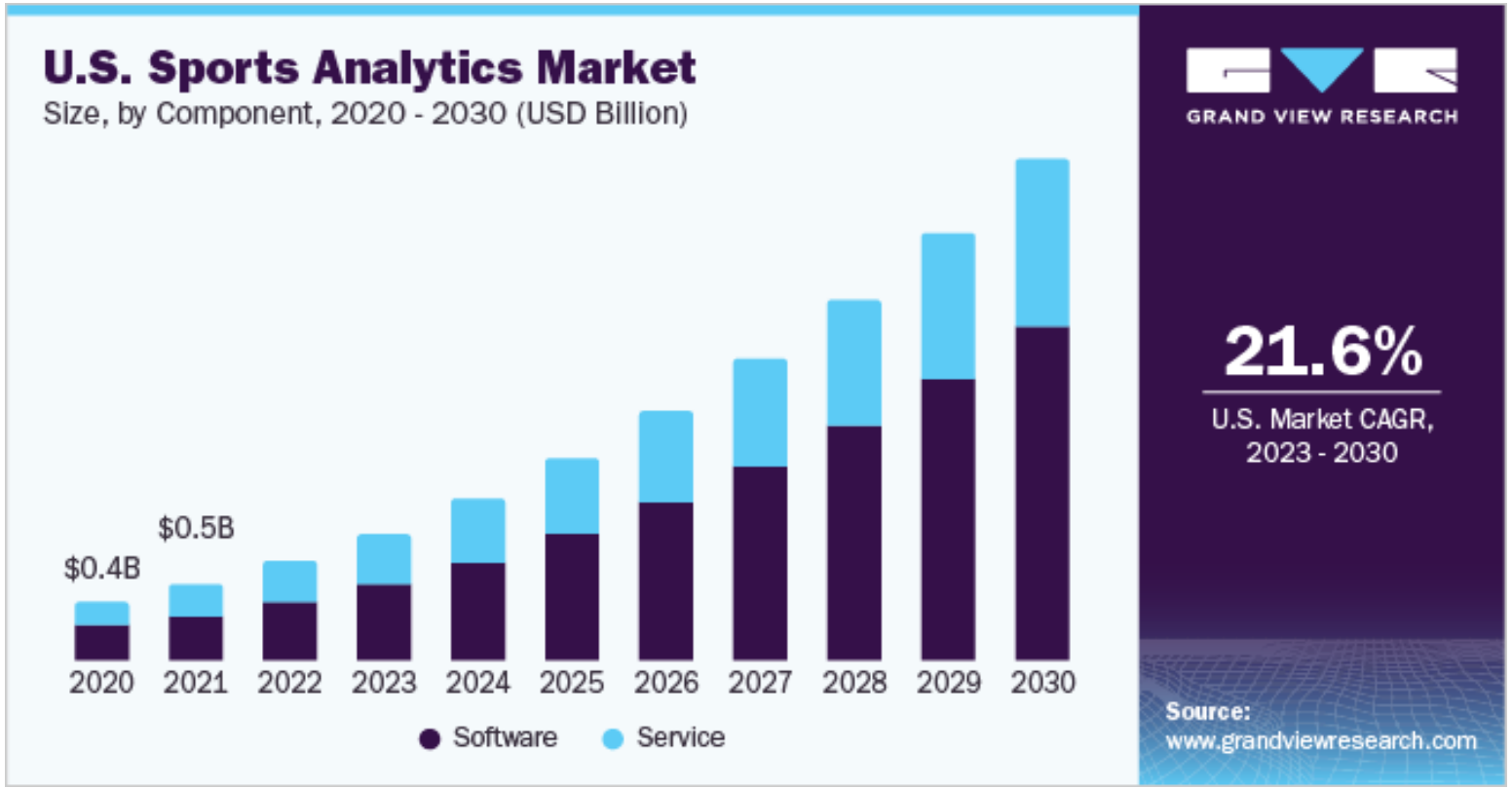
# StatChamp

강민재 | 김나현 | 지 훈

2024 딥생성 모델

# Global Sports Analytics Market




스포츠 분석 시장은 2022년 27억 3천만 달러로 평가됨

2023년부터 2030년까지 연평균 22.3% 성장할 것으로 예상

# Necessity of StatChamp

**경기 기록 및 데이터의 중요성:**

스포츠 산업에서 경기 중 측정 가능한 기록은 매우 가치가 높다.

야구, 축구에서도 선수 평가 및 우수 선수 선발에 데이터가 매우 중요하다.

**보조형 AI의 필요성:**

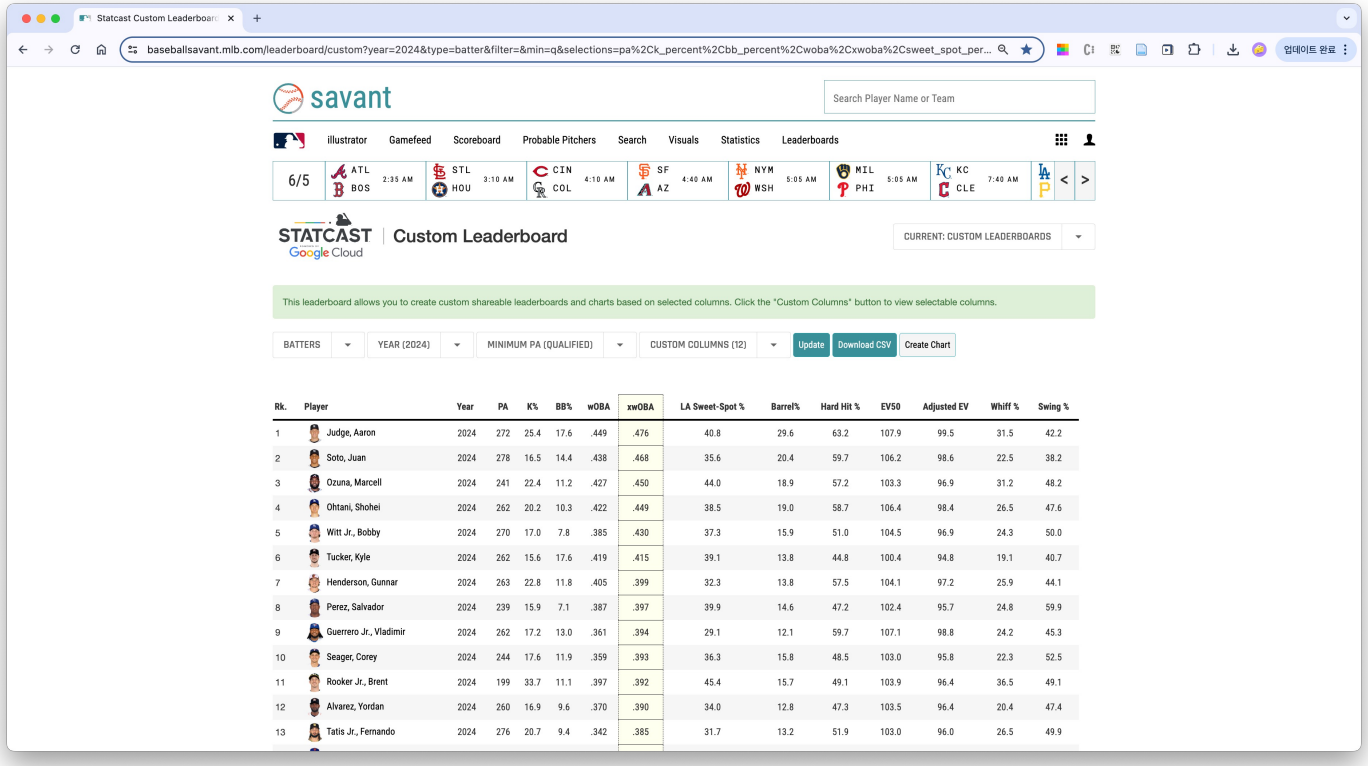
선수 기록을 수작업으로 비교하는 것은 번거로우므로 이를 돕는 보조형 AI가 필요하다.

**RAG 적용의 필요성:**

현재의 LLM 모델은 최신 스포츠 정보를 가지고 있지 못하므로 정확한 통계를 제공하지 못한다. **본 프로젝트는 축구 통계 데이터를 기반으로 LLaMA3 모델에 RAG를 적용하여 복잡한 스포츠 데이터들을 효과적으로 처리하고, 의미 있는 통찰을 도출할 수 있는 통계 기반 스포츠 문답 시스템인 Statchamp를 개발하였다.**

# Baseball Dataset

야구 데이터셋의 경우 메이저 리그 베이스볼(MLB) 통계 사이트인 Savant에서 제공하는 데이터셋을 사용하였다

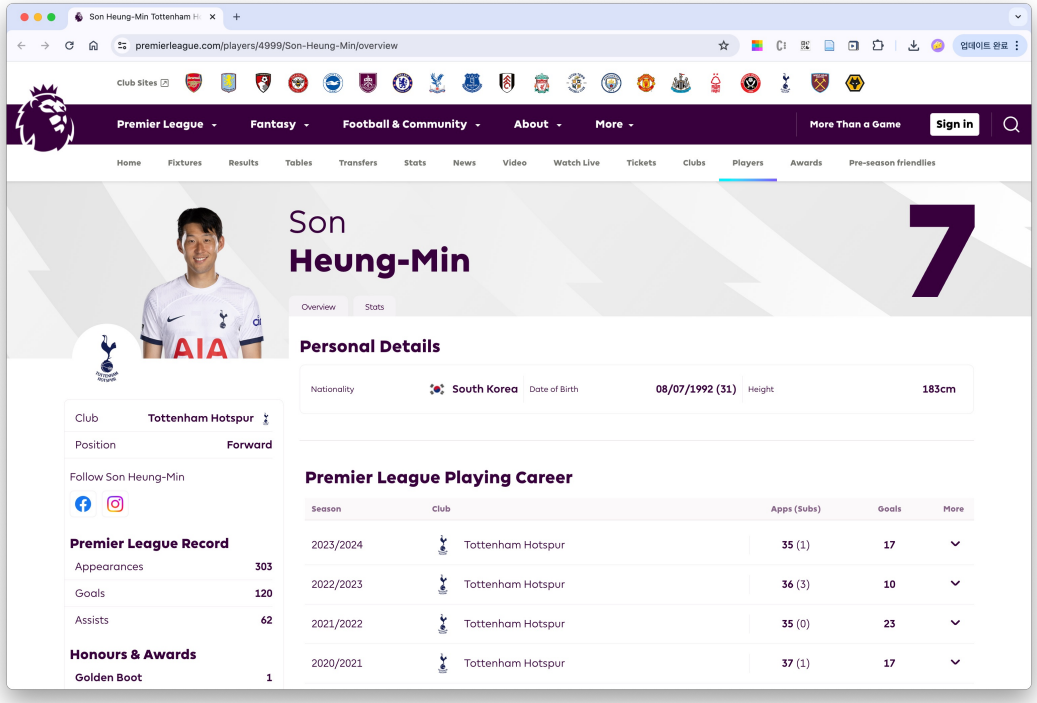



| Rk. | Player | Year | PA | K% | BB% | wOBA | xwOBA | LA Sweet-Spot % | Barrel% | Hard Hit % | EV50 | Adjusted EV | Whiff % | Swing % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Judge, Aaron | 2024 | 272 | 25.4 | 17.6 | .449 | .476 | 40.8 | 29.6 | 63.2 | 107.9 | 99.5 | 31.5 | 42.2 |
| 2 | Soto, Juan | 2024 | 278 | 16.5 | 14.4 | .438 | .468 | 35.6 | 20.4 | 59.7 | 106.2 | 98.6 | 22.5 | 38.2 |
| 3 | Ozuna, Marcell | 2024 | 241 | 22.4 | 11.2 | .427 | .450 | 44.0 | 18.9 | 57.2 | 103.3 | 96.9 | 31.2 | 48.2 |
| 4 | Ohtani, Shohei | 2024 | 262 | 20.2 | 10.3 | .422 | .449 | 38.5 | 19.0 | 58.7 | 106.4 | 98.4 | 26.5 | 47.6 |
| 5 | Witt Jr., Bobby | 2024 | 270 | 17.0 | 7.8 | .385 | .430 | 37.3 | 15.9 | 51.0 | 104.5 | 96.9 | 24.3 | 50.0 |
| 6 | Tucker, Kyle | 2024 | 262 | 15.6 | 17.6 | .419 | .415 | 39.1 | 13.8 | 44.8 | 100.4 | 94.8 | 19.1 | 40.7 |
| 7 | Henderson, Gunnar | 2024 | 263 | 22.8 | 11.8 | .405 | .399 | 32.3 | 13.8 | 57.5 | 104.1 | 97.2 | 25.9 | 44.1 |
| 8 | Perez, Salvador | 2024 | 239 | 15.9 | 7.1 | .387 | .397 | 39.9 | 14.6 | 47.2 | 102.4 | 95.7 | 24.8 | 59.9 |
| 9 | Guerrero Jr., Vladimir | 2024 | 262 | 17.2 | 13.0 | .361 | .394 | 29.1 | 12.1 | 59.7 | 107.1 | 98.8 | 24.2 | 45.3 |
| 10 | Seager, Corey | 2024 | 244 | 17.6 | 11.9 | .359 | .393 | 36.3 | 15.8 | 48.5 | 103.0 | 95.8 | 22.3 | 52.5 |
| 11 | Rooker Jr., Brent | 2024 | 199 | 33.7 | 11.1 | .397 | .392 | 45.4 | 15.7 | 49.1 | 103.9 | 96.4 | 36.5 | 49.1 |
| 12 | Alvarez, Yordan | 2024 | 260 | 16.9 | 9.6 | .370 | .390 | 34.0 | 12.8 | 47.3 | 103.5 | 96.4 | 20.4 | 47.4 |
| 13 | Tatis Jr., Fernando | 2024 | 276 | 20.7 | 9.4 | .342 | .385 | 31.7 | 13.2 | 51.9 | 103.0 | 96.0 | 26.5 | 49.9 |

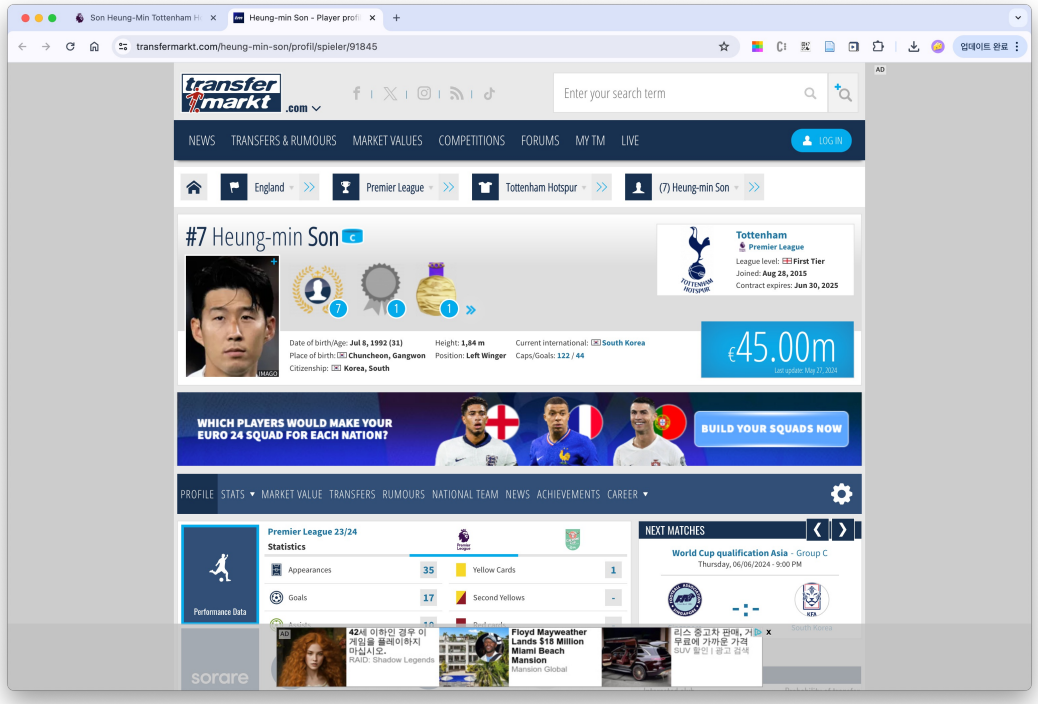
Savant

# Football Dataset

- 기존에 구할 수 있는 데이터셋의 경우 선수의 현재 시장 가치, 블로킹한 슈팅 수 등 자세한 통계 수치를 제공하고 있지 않다
- 따라서, 저명한 축구 통계 사이트인 TransferMarkt(트랜스퍼마크트)와 잉글랜드 프리미어 리그 공식 웹사이트에서 제공하는 경기 중 수집 데이터를 크롤링한다




England Premier League Official Website

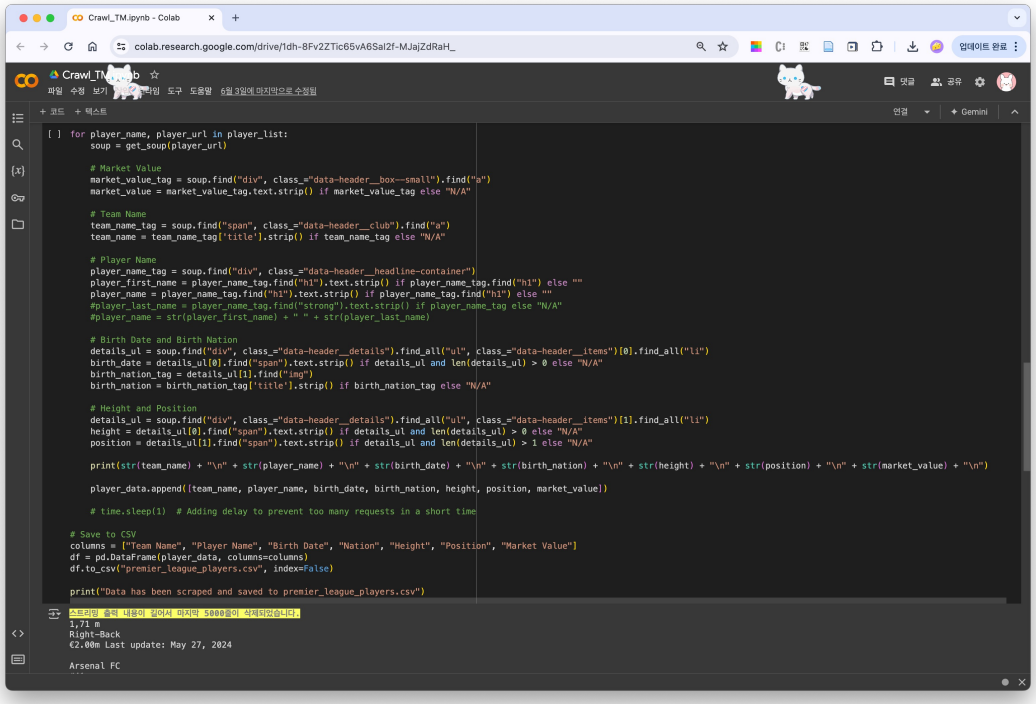



TrasnferMarkt

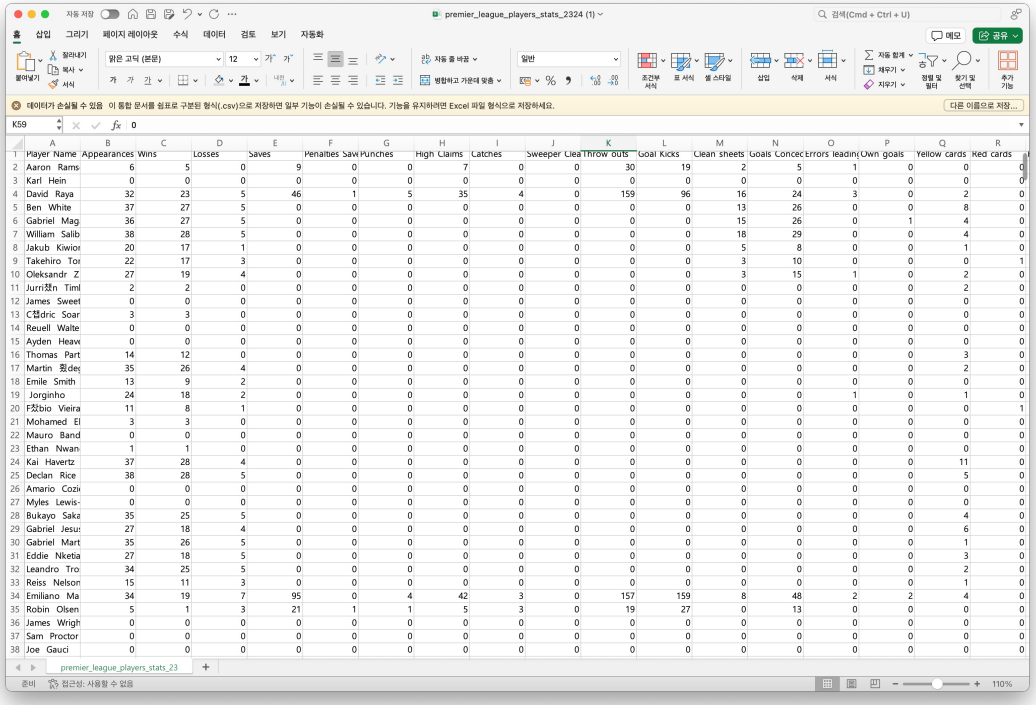
# Football Dataset Crawling

- 웹의 정적 파싱 및 동적 파싱을 통해 웹 컨텐츠를 크롤링하기 위해 파이썬 라이브러리인 BeautifulSoup과 Selenium을 사용한다

Selenium & BeautifulSoup4



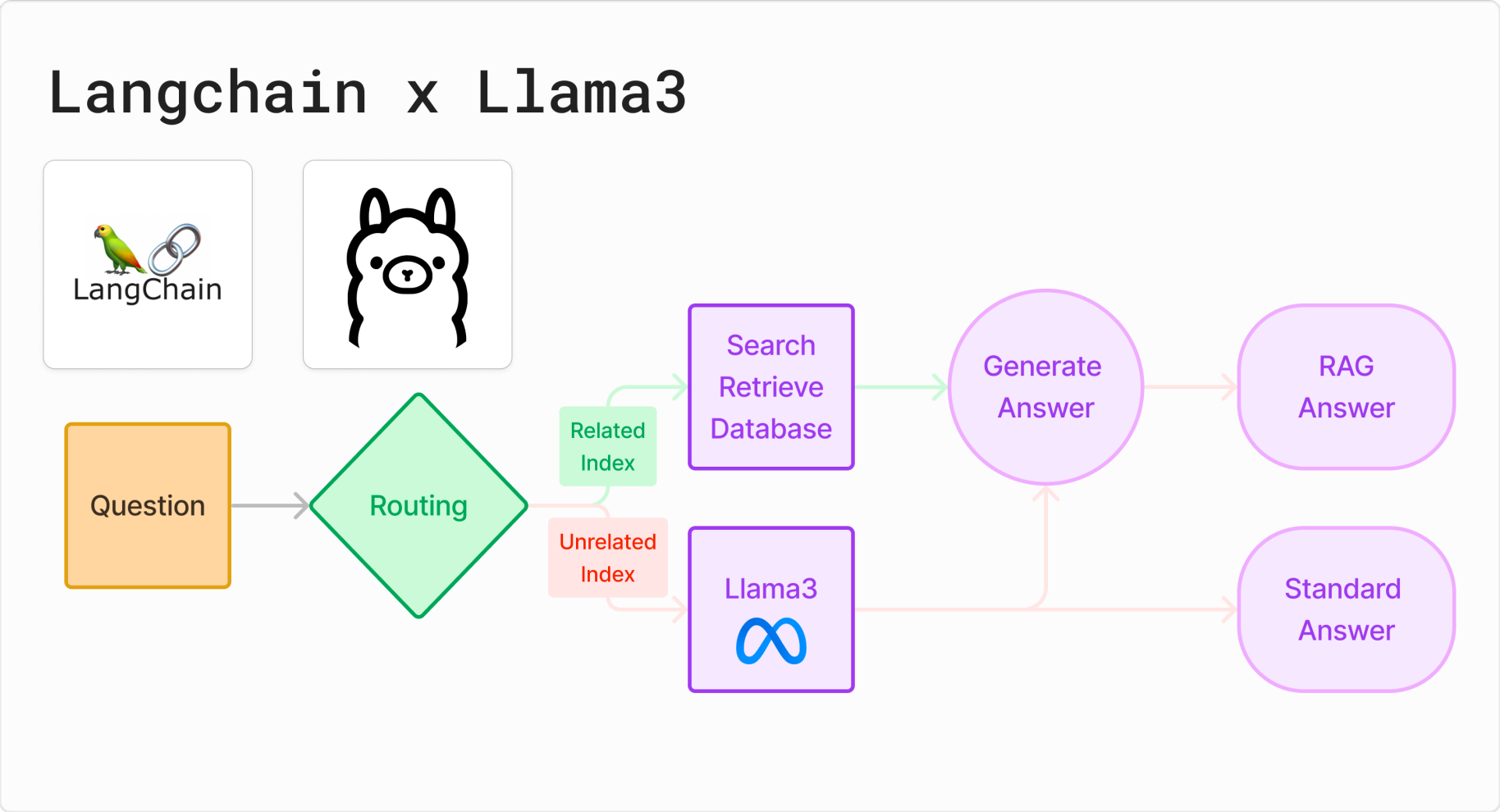

Crawling




| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Player Name | Appearances | Wins | Losses | Saves | Penalties Savi | Punches | High Claims | Catches | Sweeper Clea | Throw Outs | Goal Kicks | Clean sheets | Goals Concec | Errors leadin | Own goals | Yellow cards | Red cards |
| 2 | Aaron Rams | 6 | 5 | 0 | 9 | 0 | 0 | 7 | 0 | 0 | 30 | 19 | 2 | 5 | 1 | 0 | 0 | 0 |
| 3 | Karl Hein | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | David Raya | 32 | 23 | 5 | 46 | 1 | 5 | 35 | 4 | 0 | 159 | 96 | 16 | 24 | 3 | 0 | 2 | 0 |
| 5 | Ben White | 37 | 27 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 26 | 0 | 0 | 8 | 0 |
| 6 | Gabriel Mag | 36 | 27 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | 26 | 0 | 1 | 4 | 0 |
| 7 | William Salib | 38 | 28 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 29 | 0 | 0 | 4 | 0 |
| 8 | Jakub Kiwior | 20 | 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 8 | 0 | 0 | 1 | 0 |
| 9 | Takehiro Tor | 22 | 17 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 10 | 0 | 0 | 0 | 1 |
| 10 | Oleksandr Z | 27 | 19 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 15 | 1 | 0 | 2 | 0 |
| 11 | Jurriën Timl | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 12 | James Sweet | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | Cédric Soar | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | Reuell Walte | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | Ayden Heave | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | Thomas Part | 14 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| 17 | Martin 욌dieg | 35 | 26 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 18 | Emile Smith | 13 | 9 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | Jorginho | 24 | 18 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 20 | Fábio Vieira | 11 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 21 | Mohamed El | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | Mauro Band | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | Ethan Nwan | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | Kai Havertz | 37 | 28 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 |
| 25 | Declan Rice | 38 | 28 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 |
| 26 | Amario Cozi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | Myles Lewis- | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | Bukayo Saka | 35 | 25 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |
| 29 | Gabriel Jesu | 27 | 18 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 |
| 30 | Gabriel Mart | 35 | 26 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 31 | Eddie Nketia | 27 | 18 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| 32 | Leandro Tro | 34 | 25 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 33 | Reiss Nelson | 15 | 11 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 34 | Emiliano Ma | 34 | 19 | 7 | 95 | 0 | 4 | 42 | 3 | 0 | 157 | 159 | 8 | 48 | 2 | 2 | 4 | 0 |
| 35 | Robin Olsen | 5 | 1 | 3 | 21 | 1 | 1 | 5 | 3 | 0 | 19 | 27 | 0 | 13 | 0 | 0 | 0 | 0 |
| 36 | James Wrigh | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | Sam Proctor | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 38 | Joe Gauci | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Crawled data

# StatChamp Architecture



# Implementation

**1.데이터 로딩**

야구 및 축구 데이터를 로드하고 가공.

텍스트 형식으로 변환하여 벡터화 준비.

**2. 벡터 스토어 생성**

HuggingFace의 sentence-transformers로 데이터를 벡터화.

FAISS 인덱스 생성으로 유사도 검색 최적화.

**3. 질문 및 응답 생성**

LangChain의 ChatOllama 모델로 응답 생성.

질문과 관련성이 높은 컨텍스트를 검색해 정확한 답변 제공.

**4. 모델 초기화**

initialize 함수로 야구와 축구 모델 각각 초기화.

각 스포츠에 맞는 데이터 로드 및 응답 준비.

**5. 웹 서버 구성**

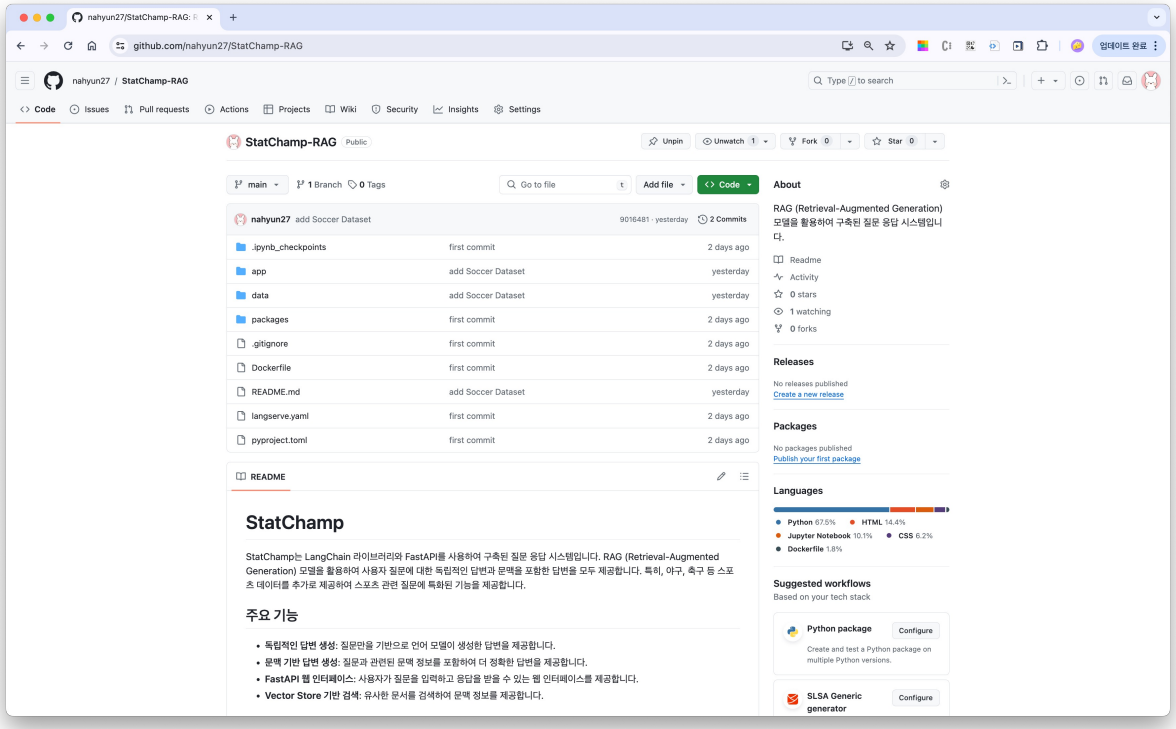
LangServe 및 FastAPI로 웹 서버 구축.

사용자가 웹 인터페이스를 통해 질문할 수 있도록 구성.

질의에 따라 야구 또는 축구 모델 로드 후 응답 생성.

```python
class RAGModel:
    def __init__(self, baseball_data_path, soccer_players_data_path, soccer_stats_data_path, soccer_stats_data_entire_path, sport):
        self.baseball_data_path = baseball_data_path
        self.soccer_players_data_path = soccer_players_data_path
        self.soccer_stats_data_path = soccer_stats_data_path
        self.soccer_stats_data_entire_path = soccer_stats_data_entire_path
        self.sport = sport
        self.data = self.load_data()
        print(f"Loaded data for {self.sport}:")
        print(self.data.head())
        self.vector_store = self.create_vector_store()
        self.llm = ChatOllama(model="llama3:latest")
        self.question_prompt = ChatPromptTemplate.from_template(
            f"You are a {self.sport} player analyst. Don't talk too hard but make it pleasant. {{data}}"
        )
        self.answer_prompt = ChatPromptTemplate.from_template(
            "Answer the question based only on the following context:\n{context}\n\nQuestion: {question}"
        )

    def load_data(self):
        if self.sport == 'baseball':
            data = pd.read_csv(self.baseball_data_path)
            data[['last_name', 'first_name']] = data['last_name, first_name'].str.split(', ', expand=True)
            data['text'] = data.apply(lambda row: f"Player: {row['first_name']} {row['last_name']}, Year: {row['year']}, PA: {row['pa']}, K%:
        elif self.sport == 'soccer':
            players_data = pd.read_csv(self.soccer_players_data_path)
            stats_data = pd.read_csv(self.soccer_stats_data_path)
            stats_data_entire = pd.read_csv(self.soccer_stats_data_entire_path)

            # 데이터 컬럼 이름 소문자 처리 및 공백 제거
            players_data.columns = players_data.columns.str.strip().str.replace(' ', '').str.lower()
            stats_data.columns = stats_data.columns.str.strip().str.replace(' ', '').str.lower()
            stats_data_entire.columns = stats_data_entire.columns.str.strip().str.replace(' ', '').str.lower()

            players_data['text'] = players_data.apply(lambda row: f"Player: {row['playername']}, Team: {row['teamname']}, Position: {row['posi
            stats_data['text'] = stats_data.apply(lambda row: f"Player: {row['playername']}, Appearances: {row['appearances']}, Goals: {row['g
            stats_data_entire['text'] = stats_data_entire.apply(lambda row: f"Player: {row['playername']}, Career Saves: {row['saves']}, Caree

            data = pd.concat([players_data[['text']], stats_data[['text']], stats_data_entire[['text']]], ignore_index=True)
        return data

    def create_vector_store(self):
        if self.data.empty:
            raise ValueError("No data available to create vector store.")  # 데이터가 비어 있을 때 예외 발생
        loader = DataFrameLoader(self.data, page_content_column='text')
        documents = loader.load()
        print("Loaded documents for vector store:")  # 디버깅 출력
        for doc in documents[:5]:  # 처음 몇 개의 문서를 출력하여 확인
            print(doc.page_content)  # 디버깅 출력
        embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
        vector_store = FAISS.from_documents(documents, embeddings)
        return vector_store

    def predict(self, chat_history, question):
        data_str = self._format_chat_history(chat_history)
        messages = [
            SystemMessage(content=f"You are a {self.sport} player analyst. Don't talk too hard but make it pleasant."),
            HumanMessage(content=data_str),
            HumanMessage(content=question)
        ]

        standalone_answer_result = self.llm.invoke(messages)
        standalone_answer = standalone_answer_result.content

        # 질문에 대한 컨텍스트 검색
        search_results = self.vector_store.similarity_search(question, k=5)
        context = "\n\n".join([result.page_content for result in search_results])
        print('\n질문과 관련있다고 판단된 참조할 컨텍스트:', search_results)

        # 컨텍스트와 질문을 사용하여 RAG 답변을 생성합니다.
        answer_messages = [
            SystemMessage(content="Answer the question based only on the following context:"),
            HumanMessage(content=context),
            HumanMessage(content=f"Question: {question}")
        ]
```

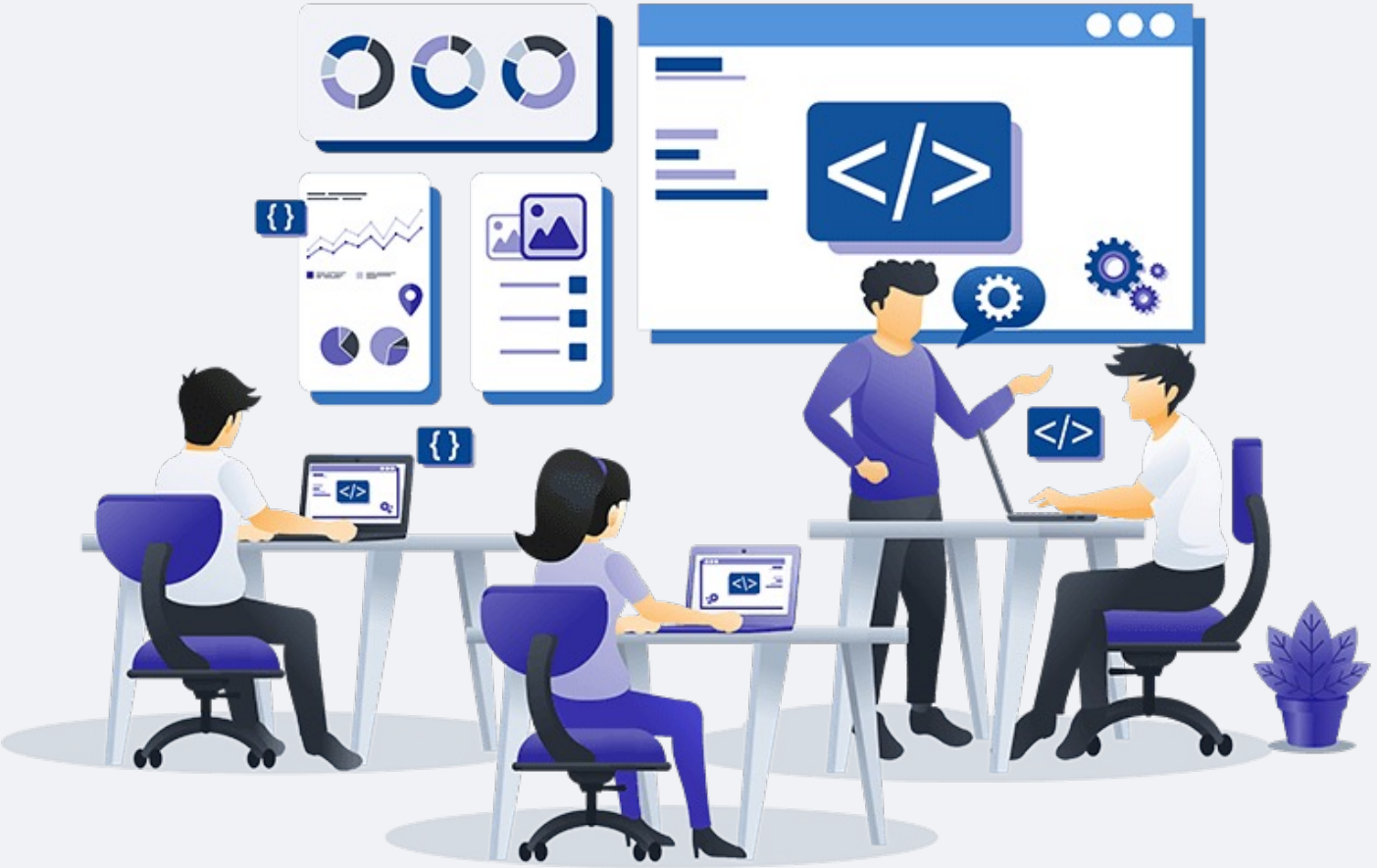
# Source Code




**코드는 아래 레포지토리에서 확인할 수 있습니다.**

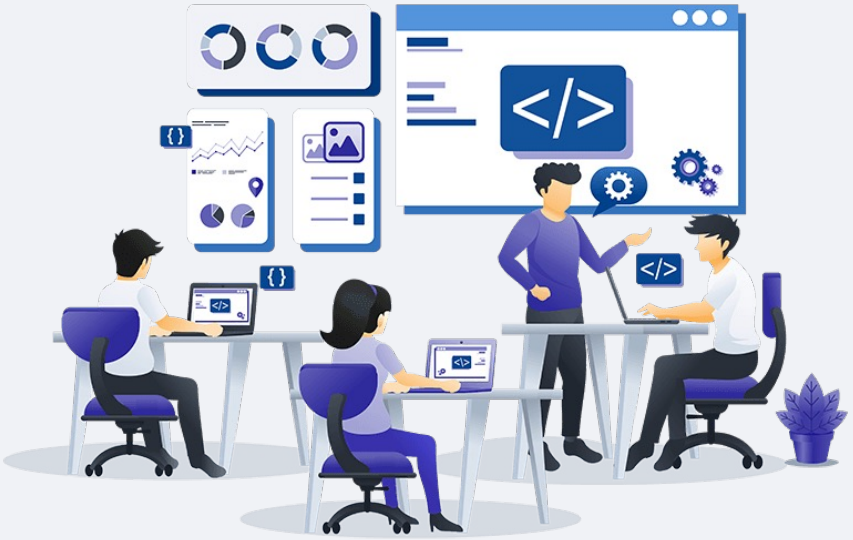
https://github.com/nahyun27/StatChamp-RAG

# Demonstration



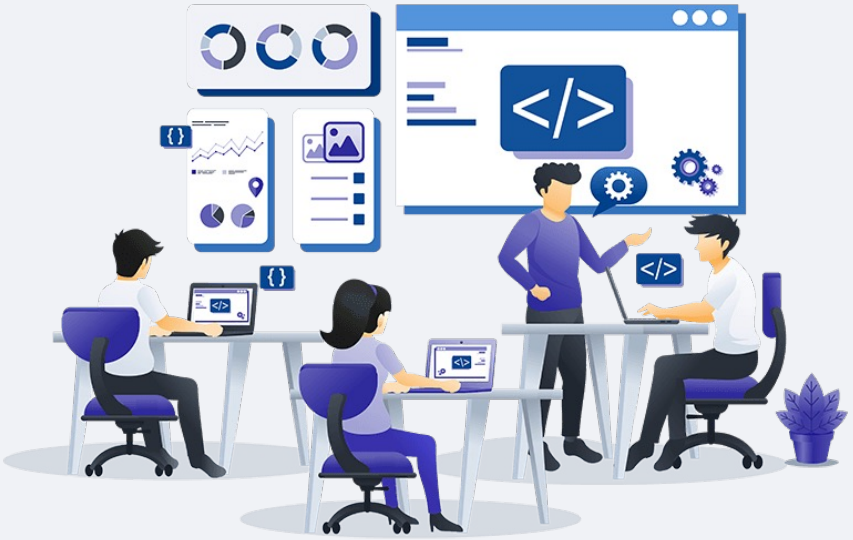
# Q1.

## Could you compare the performances of Ederson and Alisson Becker?



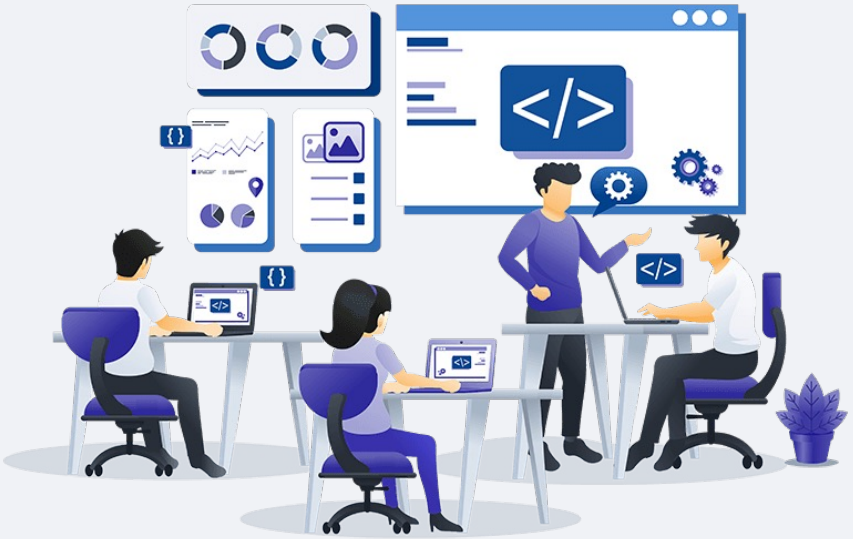
# Q2.

## How was the performance of Heung-min Son in 2023-2024 season?



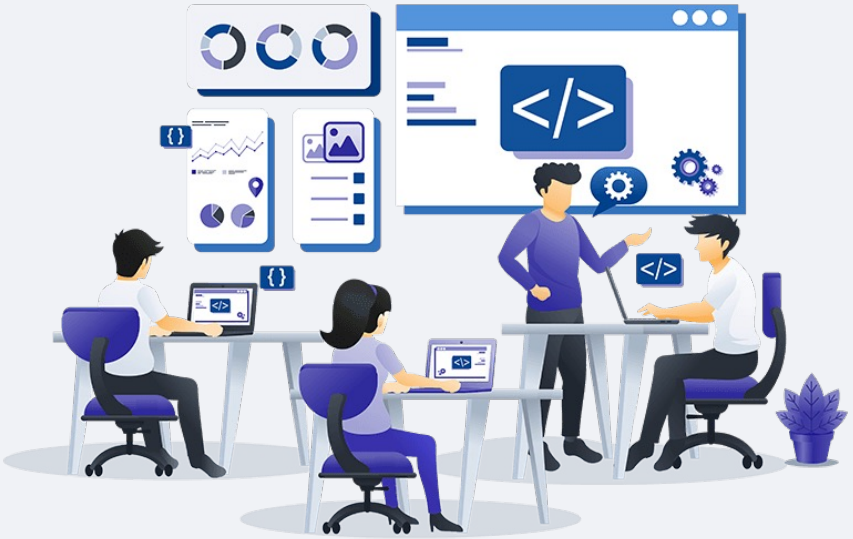
# Q3.

## How many goals has Nahyun Kim scored in the past 1 year?



# Q4.

## What is the barrel percent of the Kyle Tucker in 2024?



# 기대 효과 : 스포츠 분석가 및 애호가의 고충을 해결

각 선수마다의 기록을 수기로 비교하는 것은 매우 번거로운 일이다.

StatChamp를 통한 빠른 질의 응답으로 수작업으로 데이터를 분석에 소요되는 시간과 비용을 절감할 수 있다.

할 일이 많은 스포츠 분석가

Statchamp를 쓰고 월급 루팡이 된 스포츠 분석가

# Thank you